9-1-1992

A NEURAL NETWORK ARCHITECTURE FOR PREDICTION

W. Hsu
Purdue University School of Electrical Engineering

L. S. Hsu
Purdue University School of Electrical Engineering

M. F. TENORIO
Purdue University School of Electrical Engineering

A NEURAL NETWORK ARCHITECTURE FOR PREDICTION

W. Hsu
L. S. Hsu
M. F. TENORIO

TR-EE 92-38
SEPTEMBER 1992

SCHOOL OF ELECTRICAL ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

A Neural Network Architecture for Precliction

W. Hsu L. S. Hsu M. F. Tenorio

Parallel Distributed Structures Laboratory
Department of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

Abstract

This report describes a neural network architecture ClusNet designed for the prediction of chaotic time series. It advantages include simplicity, fast and sure convergence, and less need for computing resources. After describing its architecture and learning algorithms, its prediction performance on the Logistic and the Mackay-Class chaotic time series is presented. Compared to other current prediction approaches, ClusNet predicts with the same level of accuracy while utilizing less resources.

1 Introduction

Chaotic time series are characterized by difficulty in predicting its continuation. The approach we used to predict chaotic time series is as follows :

$$x(t+\mathrm{T}) = \mathcal{F}\{x(t), x(t-\mathrm{A}), x(t-2\Delta), \ldots, x(t-(\mathrm{m}-1)\Delta)\} \quad (1)$$

where $(x(t), x(t-\mathrm{A}), x(t-2\Delta), \ldots, x(t-(\mathrm{m}-1)\Delta))$ is called the state vector. This embedding process was first proposed by Packard et al.[1] and later given a formal treatment by Takens in [2]. The forecasting problem we address is then to represent $\mathcal{F}$ so that the value of $x(t+\mathrm{T})$ can be produced given some state vector.

There are several important chaotic time series that have been used as benchmark for prediction algorithmn. The Logistic and the Mackay-Glass are two popular ones. Many different prediction algorithms have been proposed and we describe examples from two main classes of prediction approaches below. Lapedes et al. [3] used a two hidden layers network to learn the prediction function $\mathcal{F}$. Their approach is called global because the network learns the global dynamics of the chaotic system. Training such a network is slow because of the numerical accuracy desired and the presence of two hiclden layers. A different approach is the local linear method proposed by Farmer et al. [4]. Instead of learning the global dynamics of the chaotic system as the Lapedes-Farber network, the local linear method keeps a file of thousands of previous input vectors (Eqn 1). When a new input vector $x(t)$ is presented as a basis for a prediction of $x(t+T)$, the closest vectors in the file is searched and a linear interpolation is done to the future of these from which the prediction $x(t+\mathrm{T})$ is made. All the recent development in the class of local methods e.g. [5, 6, 7] are instance-based and need large amount of storage and processing time.

In this report, we designed a **ClusNet** architecture and show how it can be used on the problem of predicting chaotic time series with good accuracy while at the same time using comparatively less computing resources than the instance-based approaches.

The organization of the paper is as follows : Section 2 describes the ClusNet architecture and its learning algorithm. Section 3 explains how the trained ClusNet can be used in prediction tasks. Section 4 details experimental results and comparisons with previous methods on predicting the Logistic and the Mackay-Glass time series. The appendices carry a proof of the convergence of ClusNet and an analysis on the number of cluster centers required.

2 The Clustering Network

The clustering network ClusNet is a neural network that classifies a given set of n input vectors into N clnsters. During the learning phase, the learning algorithm determines N weight vectors which can be interpreted as the centroids of the vectors in the clusters. It does this by minimizing the total Euclidean distances of the vectors from their respective centroids. During the prediction phase, a test vector is presented and the network determines which cluster the test vector belongs.

The next subsections describe the network architecture and the learning algorithm.

2.1 Network Architecture

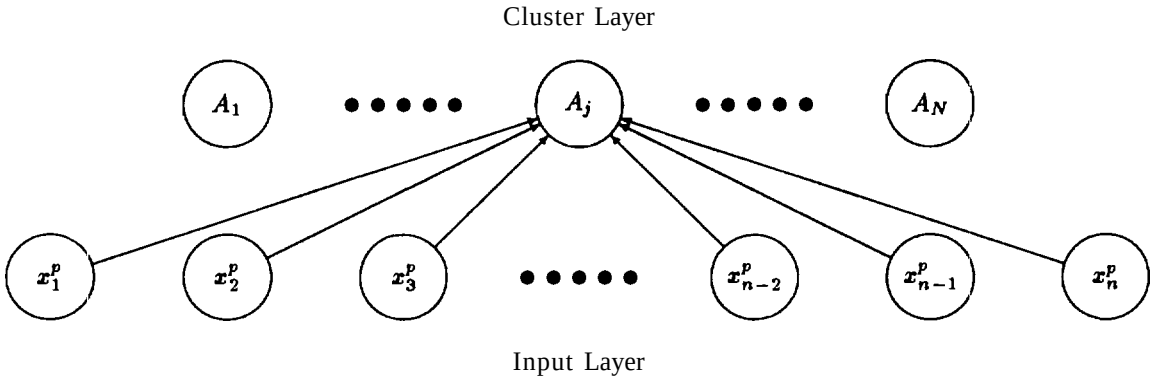

Figure 1: The ClusNet Architecture

The network consists of 2 layers as shown in Figure 1. The first layer is the input layer. It is a static layer with a fixed number of nodes. This number is equal to the length of the state vector. The nodes in this layer are referred to as input nodes.

The second layer is the cluster layer. It is a dynamic layer where nodes are dynamically created in response to data. The number of nodes corresponds to the number of clusters needed to classify the data to within a given accuracy. The nodes in this layer are called cluster nodes.

The two layers are fully connected, but there is no connection between nodes of the same layer. The weights that connects a given cluster node c to all the input nodes form the components of the weight vector W^c. The values of the weight vectors are determined by the learning algorithm which will be described in the next subsection.

Let i be an index that runs over the input nodes and c be a cluster node. When test vector $\mathbf{X}^p$ is presented, the activation at node j is defined as

$$A_c = (\mathbf{X}^p - \mathbf{W}^c)^2 \quad (2)$$

$$= \sum_i (X_i^p - W_i^c)^2 \quad (3)$$

where the square in equation (2) stands for the scalar product of the vector with itself. The cluster nodes goes through a competition and the one with the smallest activation wins. The winning unit outputs a one to the output node and all the losing nodes output zeroes.

$$O_i = \begin{cases} 1 & \text{if i wins} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

2.2 Learning Algorithm

The learning is done in two stages. During the first stage, input vectors are presented and the network creates m nodes in the second layer to group them into m clusters. The weights are changed while new vectors are being added to a cluster. This stage requires one pass of the input vectors. By the end of this pass, a vector which has been assigned to a cluster may no longer belong to that cluster because the value of the weight vectors has been changed. When this happens, we say that the cluster assignment is incorrect, and the network is not in equilibrium.

The second stage starts with the given weights determined by the previous stage, switches vectors from one cluster to another and changes the weights appropriately. In so doing, the number of incorrect assignment decreases. If incorrect assignment exists at the end of one pass, the process is repeated in another pass. This is repeated until the network is free from incorrect assignments. In this case, the network is said to be in equilibrium.

The following sub-sections describe the learning algorithms.

2.3 Weight Determination

Learning involves deciding which input vectors belong to a cluster and the weight of that cluster. The two are adjusted at the same time. When the network converges, we know the total error is

$$E = \sum_{k=1}^{N} \sum_{p=1}^{n_k} (\mathbf{W}^k - \mathbf{X}^p)^2 \tag{5}$$

where n_k is the number of input vectors in cluster k, and N is the total number of clusters.. To minimize E, we let

$$\frac{\partial E}{\partial W_i^j} = 0 \tag{6}$$

This becomes

$$\sum_{p=1}^{n_j} (\mathbf{W}^j - \mathbf{X}^p) = 0 \tag{7}$$

therefore

$$\mathbf{W}^j = \frac{1}{n_j} \sum_{p=1}^{n_j} \mathbf{X}^p \tag{8}$$

This shows that the weights can be determined if we know the input vectors which belong to the cluster. Subsequent subsections describe how the clustering is decided upon.

2.4 Learning algorithm for Stage 1

We start off by creating a class node C_1 and assign the following weight vector to it.

$$\mathbf{W}^1 = \mathbf{X}^1 \tag{9}$$

Next, we assume that k class nodes have been created, and an input vector $\mathbf{X}^p$ is presented to the network. We compute the activation of all existing class nodes from 1 to k, which are denoted by:

$$A_1, \ldots, A_k.$$

Now if node m has the smallest activation A_m and this value is less than a predefined constant ϵ, we include this new input vector in cluster m by adjusting the weight vector for that cluster

$$\mathbf{W}^m = \frac{1}{n_m} \{(n_m - 1)\mathbf{W}^m + \mathbf{X}^p\} \tag{10}$$

where n_m is the number of vectors captured in cluster m, after the new vector $\mathbf{X}^p$ bas been added.

On the other hand, if A_k is not less than ϵ, we create a new class node $k+1$, and set the weight vector to be equal to the input vector:

$$\mathbf{W}^{k+1} = \mathbf{X}^p \tag{11}$$

This is done until all the input vectors in the' training set

$$\mathbf{X}^1, \ldots, \mathbf{X}^n.$$

are exhausted.

2.5 Learning algorithm for Stage 2

During stage 1, cluster nodes were created and input vectors were provisionally assigned to one of the clusters. The weight vectors were adjusted during this process. At the end of that stage, an input vector $\mathbf{X}^p$ which was assigned to cluster c may become closer to the weight vector of cluster d. When this happens, we say that the assignment is incorrect. In this case, the total error

$$E = \sum_{k=1}^{N} \sum_{p=1}^{n_k} (\mathbf{W}^k - \mathbf{X}^p)^2 \tag{12}$$

is not minimum.

Let $A_1, \ldots, A_N$ be the activation of the cluster nodes when $\mathbf{X}^p$ is presented, and let the weight vectors of the nodes be $\mathbf{W}^1, \ldots, \mathrm{W}^N$. The above condition means that

$$A_c = (\mathbf{W}^c - \mathbf{X}^p)^2 \tag{13}$$

is greater than

$$A_d = (\mathbf{W}^d - \mathbf{X}^p)^2 \tag{14}$$

To overcome this, we transfer the vector $\mathbf{X}^p$ from cluster c to cluster d by adjusting the weights as follows:

$$\mathbf{W}'^c = \frac{1}{n_c - 1}\{n_c \mathbf{W}^c - \mathbf{X}^p\} \tag{15}$$

$$\mathbf{W}'^d = \frac{1}{n_d + 1}\{n_d \mathbf{W}^d + \mathbf{X}^p\} \tag{16}$$

This process is repeated until there is no incorrect assignments. The fact that the process converges will be proved in Appendix A.

3 Prediction Application

We shall now make use of the network described above to predict the temporal continuation of a time series. We shall assume that embedding has been done. Then the learning phase consists of presenting the pairs :

$$(\mathbf{X}^i, \mathbf{Y}^i) \quad \mathrm{I} \leq i \leq \mathrm{n} \tag{17}$$

to ClusNet. The prediction task then is to return the correct value of $\mathbf{Y}^p$ given the state vector $\mathbf{X}^p$. Typically, $\mathbf{X}^i$ has several components, i.e.

$$\mathbf{X}^i = [X_1^i, X_2^i, \ldots, X_d^i] \tag{18}$$

where each X_j^i maybe a delayed sample of the time series or other independent indicator. For simplicity, we will restrict the following discussion to the case where Y is scalar only.

We assume that there exist a function $\mathcal{F} : R^d \rightarrow R^1$ such that

$$\mathbf{Y} = \mathcal{F}(\mathbf{X}) \tag{19}$$

We choose a set of $\mathbf{X}$ which belong to cluster i. A Taylor series expansion up to the linear term at $\overline{\mathbf{X}}$ (the cluster center of cluster i) is

$$Y = \mathcal{F}(\overline{\mathbf{X}}) + (X - \overline{X}) \cdot \nabla \mathcal{F}(\overline{\mathbf{X}}) \tag{20}$$

It is trivial to show that the average of Y is

$$\overline{\mathbf{Y}} = \mathcal{F}(\overline{\mathbf{X}}) \tag{21}$$

The function $\mathcal{F}$ is unknown. Therefore, its gradient $\nabla \mathcal{F}$ is also unknown. We write the expansion as

$$Y = \overline{\mathbf{Y}} + (\mathbf{X} - \overline{\mathbf{X}}) \cdot \Omega \tag{22}$$

where Ω is a column vector whose components are parameters, i.e.

$$\Omega = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_d \end{pmatrix} \tag{23}$$

where d is the size of the input layer of ClusNet. The ω_i's can be determined for each cluster using linear algebra techniques.

3.1 Zero Order Prediction

In this simple prediction scheme, the ω_i's are set to zero.

3.2 First Order Prediction

In this precliction scheme, the prediction are influenced by the difference between $\mathbf{X^p}$ and the cluster centers $W^i, 1 \leq i \leq N$. The coefficient vector Ω^k for cluster C are computed as follows :

$$(Y^i - \overline{Y}) = \Omega^k \cdot (X^i - \overline{X}) \tag{24}$$

where $X^i, Y^i \in$ cluster k. Let n_k be the number of samples in cluster C. If $n_k = d$, Ω^k is unique. If $n_k >$ d, the system is overdetermined and Ω^k can be determined using least squares with singular value decolnposition. If $n_k < d$, R^k is set to a zero vector of the appropriate size.

3.3 Robustness of ClusNet

When $\mathbf{X}^p$ is presented to ClusNet for prediction where

$$\|\mathbf{X}^p - \mathbf{W}^i\| > \epsilon \quad 1 \leq i \leq N, \tag{25}$$

a special value η is generated to signify that no training samples similar to $\mathbf{X}^p$ has been seen in training and thus no prediction is possible.

4 Analysis of Space and Time Requirements

There are two major advantages in using ClusNet for prediction. It reduces memory and increases speed.

First let us look at the space requirement. Let there be n input vectors of length l, the instance space moclel requires $n * (l + 1)$ words to keep track of all the instances. In the ClusNet model, the input vectors are classified into N clusters. We need only keep the information of the m weights and the average value of the predictions for each cluster. The number of words needed is $N * (1 + 1)$. Thus there is a n/N fold saving in space requirement.

Next, let us consider the time requirement. The instance space model requires the computation of n Euclidean distances to make a prediction while the **ClusNet** model needs only m such computations. The **saving** is again of the order n/N.

The **factor** n/N depends on the degree of smoothness of the underlying function **F.** As the **smoothness** of **F** decreases, the number of cluster centers, N, required to represent it with a fixed accuracy ϵ increases with a rate proportional to $\sqrt{C}$. This analysis is carried out for a one dimensional problem **in** Appendix B.

5 Empirical Results

To compare the predictive power of different algorithms, the normalized root mean **square** error (nrmse) is used. Given a set S of pairs of the actual values (or targets, y_k) and predicted **values** ($\hat{y}_k$),

$$nrmse(S) = \sqrt{\frac{\sum_{k \in S}(target_k - prediction_k)^2}{\sum_{k \in S}(target_k - mean)^2}} \quad (26)$$

$$= \sqrt{\frac{1}{\hat{\sigma}^2}\frac{1}{N}\sum_{k \in S}(y_k - \hat{y}_k)^2} \quad (27)$$

The **averaging** (division by N, the number of data points in a set S) makes the measure independent of the size of the size. The normalization (division by $\hat{\sigma}^2$, the estimated variance of the data) removes the dependence on the dynamic range of the data. This normalization implies that if the estimated mean of the data is used as predictor, $nrmse = 1.0$ is obtained.

5.1 Application 1 : The Logistic Equation

The Logistic or Feigenbaum equation is generated by

$$x(t+1) = 4b \cdot x(t)\{1 - x(t)\} \quad (28)$$

where b = 1.0. Using ClusNet with $\epsilon = 0.0001$, we obtained Table 1. The best prediction is obtained **when** the number of cluster $m = 50$ for this problem.

Num Clusters	nrmse	E	n
10	0.010	0.3840	6
20	0.0028	0.0912	8
30	0.0013	0.0420	3
40	0.0007	0.0189	3
50	0.0004	0.0108	3
60	0.0083	0.0076	3
62	0.0083	0.0071	3

Table 1: Prediction Accuracies on the Logsitic Equation. The maximum number of clusters obtained with $\epsilon = 0.0001$ is 62.

5.2 Application 2 : The Mackay-Glass Equation

The **Mackey-Glass** equation was first proposed as a model of white blood cell **production[8]** and subsequently popularized in the non-linear field due to its richness in **structure[9].** It is a time-delay differential equation, namely

$$\frac{\partial x}{\partial t} = \frac{ax(t-\Delta)}{[1 + x^c(t-\Delta)]} - bx(t), \quad (29)$$

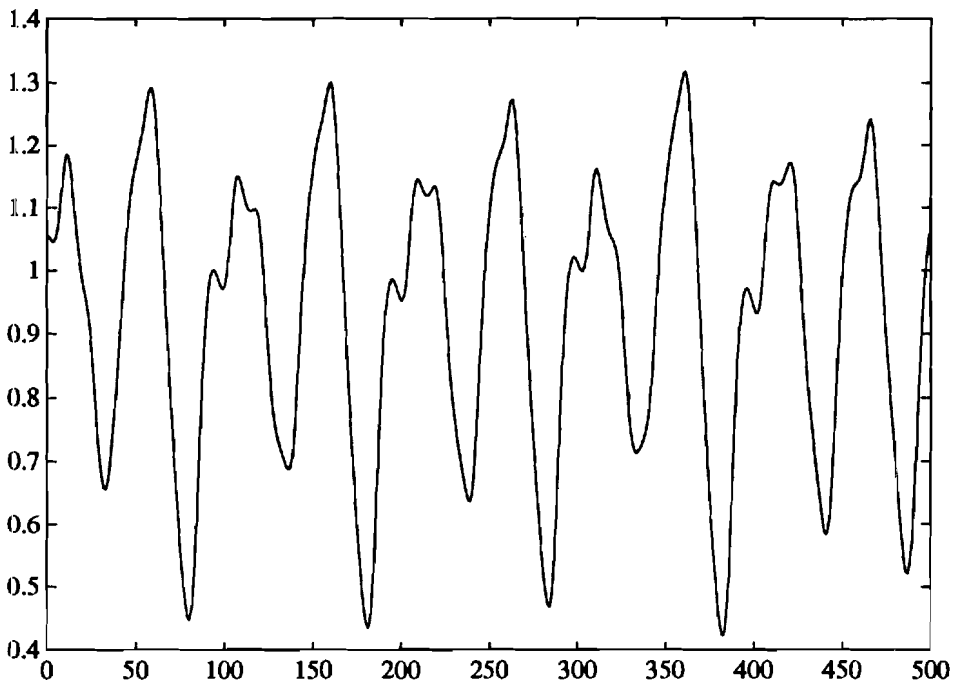

Figure 2: Plot of 500 points of the Mackey-Glass Equation with A = 17

where the constants are taken to be a = 0, 2, b = 0.1 and c = 10. The delay parameter A determines the nature of the chaotic behavior displayed by the time series. A sample of the equation is plotted in Figure 2. This series with A = 17 has been a benchmark in much of neural network research communities. The behavior of the M-G equation as a function of A is studied in [9] and tabulated in Table 2. At A = 17, x(t) appears to be quasiperiodic and the power spectrum is broadband

A < 4.53	a stable fixed point attractor
4.53 < A < 13.3	a stable limit cycle attractor
13.3 < A < 16.8	period of limit cycle doubles
A > 16.8	chaotic attractor characterized by A

Table 2: The Mackey-Glass Equation as a function of A

with numerous spikes due to the quasiperiodicity. The characteristic time of this time series is ≈ 50 steps. The standard embedding for the Mackey-Glass time series with A = 17 is $m = 4$ and sampling interval $\tau = 6$. These parameters are also used in this experiment.

Method	Training Set Size	T = 6	T = 85
Cascade Correlation	500	0.06	0.32
Back-Prop	500	0.02	0.05
6th order polynomial	500	0.04	0.85
linear predictive method	2000	0.55	0.60
SONN (Tenorio)	100	0.08	-
RBF (Casdagli)	500	0.01	-
Weighted Maps (Stokbro)	500	0.013	-
Local linear (Farmer)	1500	0.010	0.084
Linear Interpolate (Linsay)	1500	0.064	0.440
Nearest Neighbor	500		0.192
SBF 1 (Hsu)	1500	0.089	0.158
ClusNet (Hsu)	500	0.014	0.136

Table 3: Normalized Root Mean Square Prediction Error on the Mackay-Glass A = 17. Size of prediction set = 500

Number of Cluster	T=85 nrmse	T=6 nrmse	E	n
10	0.30	0.055	11.23	10
20	0.19	0.0299	5.0	13
30	0.16	0.0177	3.0	11
40		0.0138		
50	0.14	0.0162	1.5	3
55	0.136	0.0155	1.37	3

Table 4: Performance of ClusNet on the Mackay-Glass time series T = 85 and T = 6 . Training set size = 500. Prediction set size = 500. $\epsilon = 0.01$. E represents the cumulative error of the training samples with repsect to the cluster centers. n is the number of iterations required for the learning algorithm to converg on the training samples. The maximum number of clusters with $\epsilon = 0.01$ is 55 for this problem

The first 4 lines of Table 3 is cited from [10]. SONN is a self organizing neural network that constructs a global model of the time series from the training set[11]. SONN was trained on 100

points and predicted 400 points following the training set. Casdagli used radial basis functions (RBF) to fit the training set. In his implementation, he used as many RBF's as there are data points[12]. The results for Weighted Maps are taken from [13]. The results for the various algorithms marked by (*) are obtained using the same set of data. **SBF 1** [7] predicts the time series using a moving window of training samples. The configuration of **ClusNet** used here has 4 input nodes and the parameter $\epsilon = 0.01$ resulting in 55 class nodes for the 500 training samples. For this prediction problem, **ClusNet** uses an order of magnitude less storage as well as processing time to make the 500 predictions compared to **SBF 1** and other instance based methods e.g. the RBF and the local linear methods. The accuracies thus obtained are comparable if not better than most other prediction techniques for this problem.

In Table 4, we show the variation of the number of clusters on the prediction accuracies obtained. As the number of cluster increases, we see a increase in the prediction accuracies of ClusNet. Above a critical threshold (in this case 60), increasing the number of clusters does not improve the accuracies. The prediction results are much better for the T = 6 case and its critical threshold occur at about 40 cluster centers. It is interesting to note that as the number of clusters becomes the number of training samples, **ClusNet** describes the instance-base algorithms.

6 Summary

The Clustering Network (ClusNet) is novel in the following aspects : It is an on-line learning procedure by design. As a result, it does not require all the training samples to be present at training time but is able to improve its prediction as more data are presented. To be able to do this, it dynamically creates the necessary number of clusters nodes in response to the problem. To make the comparison among different prediction algorithms easy, ClusNet was used in a batch prediction mode. Its performance as a on-line method will be discussed in another paper.

Another interesting feature about ClusNet is that unlike other clustering algorithm e.g. the Kohonent network where the convergence of the algorithm can only be established empirically, **ClusNet** is guaranteed to converged and do so reasonably quickly in empirical studies. Furthermore, ClusNet is robust in that it refrain from making predictions if no similar enough examples have been seen.

References

[1] N. H. Packard. Geometry from a time series. Physics Review Letters, 1980.

[2] F. Takens. Detecting strange attractors in turbulence. In D. A. Rand and L. S. Young, editors, *Dynamical systems* and turbulence. Spinger, 1981.

[3] A. Lapedes and Farber R. Nonlinear signal processing using neural networks: prediction and system modelling. Technical Report LA-UR-87-2662, Los Alamos National Lab, 1987.

[4] D. Farmer and J. J. Sidorowich. Predicting chaotic time series. Physics Review Letters, pages 845–848, 1987.

[5] P. Linsay. An efficient method of forecasting chaotic time series using linear interpolation. Physics Letters A, pages 353–356, 1991.

[6] W. Hsu and M. F. Tenorio. Plastic network for predicting the mackey-glass time series. In *ICJNN* (Baltimore), pages 941–946, 1992.

[7] W. Hsu and M. F. Tenorio. A similarity-based approach to forecasting. In Workshop on Neural Networks (WNN), 1992.

[8] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. Science, 197:287, 1977.

[9] D. Farmer. Chaotic attractors of an infinite-dimensional dynamical system. *Physica 4D*, pages 366–393, 1982.

[10] R. S. Crowder. Predicting the mackey-glass timeseries with casecade-correlation learning. In D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton, editors, *Proc* of the 1990 *Summer* School of Connectionist Models, pages 117–123. Morgan Kaufmann, 1990.

[11] M. F. Tenorio and W. T. Lee. Self organizing network for optimum supervised learning. IEEE *Transactions* on Neural Networks, 1(1):100–110, March 1990.

[12] M. Casdagli. Nonlinear prediction of chaotic time series. Physica D, pages 335–356, 1989.

[13] K. Stokbro and D. K. Umberger. Forecasting with weighted maps. In M. Casdagli and S. Eubank, editors, Nonlinear modeling and forecasting, pages 73–93. Addison-Welsey, 1091.

Appendix A

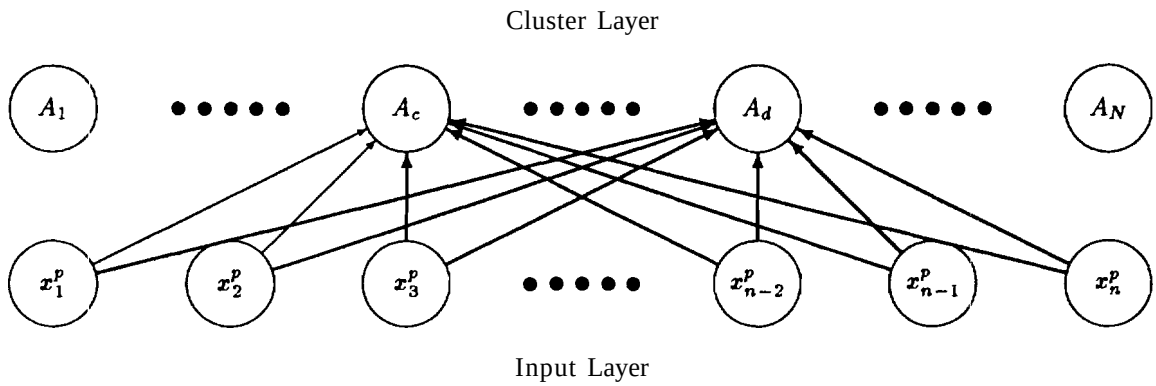

Figure 3: ClusNet Architecture

Consicler the network shown in Figure 3. There are N nodes in the cluster layer. It is being trained with the set of input vectors

$$\mathbf{X}^1, \ldots, \mathbf{X}^n.$$

After the first p − 1 input vectors are considered, the value of the weight vectors are

$$\mathbf{W}^1, \ldots, \mathbf{W}^N.$$

and the population of the clusters are

$$N_1, \ldots, n_N.$$

At this point, the next vector $\mathbf{X}^p$ from the input set, which has been allocated to cluster c, is presented. Using the above weight vectors, the activations are found to be

$$A_1', \ldots, A_N'.$$

Suppose A_c' is the minimum, the original assignment is correct, and we go on to the next vector $\mathbf{X}^{(p+1)}$. If A&is the minimum, we have

$$\mathrm{A\&} < A_c'$$

which means

$$(\mathbf{W}^d - \mathbf{X}^p)^2 < (\mathbf{W}^c - \mathbf{X}^p)^2 \tag{30}$$

In this case, we have to re-allocate the vector $\mathbf{X}^p$ from cluster c to cluster d. This involves the change in the weight vectors of these two clusters. Let the new vectors be $\mathbf{W}'^c$ and $\mathbf{W}'^d$ respectively, then we have

$$\mathbf{W}'^c = \frac{1}{n_c - 1} \sum_{k=1, k\neq p}^{n_c} \mathbf{X}^k \tag{31}$$

$$\mathbf{W}'^d = \frac{1}{n_d + 1} \{ \sum_{k=1}^{n_d} \mathbf{X}^k + \mathbf{X}^p \} \tag{32}$$

We note that the corresponding old weight vectors are

$$\mathbf{W}^c = \frac{1}{n_c} \sum_{k=1}^{n_c} \mathbf{X}^k \tag{33}$$

$$\mathbf{W}^d = \frac{1}{n_d} \sum_{k=1}^{n_d} \mathbf{X}^k \tag{34}$$

With the new weight vectors, the values of activation of the two affected clusters are

$$A'_c = \sum_{k=1, k\neq p}^{n_c} (\mathbf{W}'^c - \mathbf{X}^k)^2 \tag{35}$$

$$A'_d = \sum_{k=1}^{n_d} (\mathbf{W}'^d - \mathbf{X}^k)^2 + (\mathbf{W}'^d - \mathbf{X}^p)^2 \tag{36}$$

We note that with the old weight vectors, the corresponding values were

$$A_c = \sum_{k=1, k\neq p}^{n_c} (\mathbf{W}'^c - \mathbf{X}^k)^2 + \frac{n_c}{n_c - 1} (\mathbf{W}^c - \mathbf{X}^p)^2 \tag{37}$$

$$A_d = \sum_{k=1}^{n_d} (\mathbf{W}'^d - \mathbf{X}^k)^2 - \frac{n_d}{(n_d + 1)^2} (\mathbf{W}^d - \mathbf{X}^p)^2 \tag{38}$$

Simple algebraic manipulation gives

$$A'_c - A_c = \frac{n_c}{n_c - 1} (\mathbf{W}^c - \mathbf{X}^p)^2 \tag{39}$$

$$A'_d - A_d = \frac{n_d}{n_d + 1} (\mathbf{W}^d - \mathbf{X}^p)^2 \tag{40}$$

We define T as the total activation of all the cluster nodes

$$T = \sum_{k=1}^{N} A_k \tag{41}$$

then since the activation of the rest of the nodes are unchanged, the change in T is given by

$$\Delta T = A'_c + A'_d - A_c - A_d \tag{42}$$

which turru out to be

$$\Delta T = \frac{n_d}{n_d + 1} (\mathbf{W}^d - \mathbf{X}^p)^2 - \frac{n_c}{n_c - 1} (\mathbf{W}^c - \mathbf{X}^p)^2 \tag{43}$$

Now since

$$\frac{n_d}{n_d+1} < \frac{n_c}{n_c-1} \tag{44}$$

and

$$(\mathbf{W}^d - \mathbf{X}^p)^2 < (\mathbf{W}^c - \mathbf{X}^p)^2 \tag{45}$$

we have

$$\Delta T < 0 \tag{46}$$

which means that T forms a *monotonically decreasing* sequence. Since $T > 0$, the iteration will stop when the minimum value of T is reached.

Appendix B

For illustrative simplicity, consider a mapping from $R^1 \rightarrow R^1$. Choose coordinates in the domain so that the inputs lie in the unit interval [0,1]. Let the nonlinear mapping from [0,1] to R^1 be denoted by f and the value at a point $x \in [0,1]$ by $f(x)$. By choosing the approximation to be equal to f at N cluster centers : $x_1 = 0, x_2, \ldots, x_{N-1}, x_N = 1$, the approximate function f can be defined between these cluster centers by linear interpolation. If $x = i \leq x \leq x_{i+1}$, then

$$\hat{f} = f(x_i) + (x - x_i)\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \tag{47}$$

Next, we cliscuss how to choose N and x_i so as to keep the error below ϵ.

This is intuitive related to the rate of change of the function over the range of input values. Assume the rate of change, i.e. the second derivative of the map, is bounded by a constant C.

The maximum error in approximating a function f whose second derivative

$$\frac{\partial^2 f}{\partial x^2} \leq C \tag{48}$$

is given by

$$e(x) = -\frac{C}{2}x^2 + \frac{C}{2}(x_1 + x_2)x - \frac{Cx_1x_2}{2}. \tag{49}$$

The largest error occurs at the midpoint

$$x = \frac{x_1 + x_2}{2} \tag{50}$$

and has the value

$$\frac{C}{8}(x_2 - x_1)^2. \tag{51}$$

For this error to be less than ϵ,

$$|x_2 - x_1| < \sqrt{\frac{8\epsilon}{C}} \tag{52}$$

This tells us that the largest number of cluster centers we need is

$$N = \frac{1}{|x_2 - x_1|} = \sqrt{\frac{C}{8\epsilon}} \tag{53}$$